Rain Fall Data

1.Data collection

In [79]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [80]:

```
df=pd.read_csv(r"C:\Users\Svijayalakshmi\Downloads\rainfall in india 1901-2015.csv")
df
```

Out[80]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-De
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	3373.2	136.3	560.3	1696.3	980.
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7	159.8	458.3	2185.9	716.
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	2957.4	156.7	236.1	1874.0	690.
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	3079.6	24.1	506.9	1977.6	571.
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	2566.7	1.3	309.7	1624.9	630.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4	184.3	14.9	1533.7	7.9	196.2	1013.0	316.
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9	12.4	8.8	1405.5	19.3	99.6	1119.5	167.
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8	78.1	26.7	1426.3	60.6	131.1	1057.0	177.
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2	59.0	62.3	1395.0	69.3	76.7	958.5	290.
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4	231.0	159.0	1642.9	2.7	223.9	860.9	555.

4116 rows × 19 columns

2)Data Cleaning and Preprocessing

In [81]:

```
df.head()
```

Out[81]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
0	ANDAMAN & NICOBAR ISLANDS	1901	49.2	87.1	29.2	2.3	528.8	517.5	365.1	481.1	332.6	388.5	558.2	33.6	3373.2	136.3	560.3	1696.3	980.3
1	ANDAMAN & NICOBAR ISLANDS	1902	0.0	159.8	12.2	0.0	446.1	537.1	228.9	753.7	666.2	197.2	359.0	160.5	3520.7	159.8	458.3	2185.9	716.7
2	ANDAMAN & NICOBAR ISLANDS	1903	12.7	144.0	0.0	1.0	235.1	479.9	728.4	326.7	339.0	181.2	284.4	225.0	2957.4	156.7	236.1	1874.0	690.6
3	ANDAMAN & NICOBAR ISLANDS	1904	9.4	14.7	0.0	202.4	304.5	495.1	502.0	160.1	820.4	222.2	308.7	40.1	3079.6	24.1	506.9	1977.6	571.0
4	ANDAMAN & NICOBAR ISLANDS	1905	1.3	0.0	3.3	26.9	279.5	628.7	368.7	330.5	297.0	260.7	25.4	344.7	2566.7	1.3	309.7	1624.9	630.8

In [82]:

```
df.tail()
```

Out[82]:

	SUBDIVISION	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	Jan-Feb	Mar-May	Jun-Sep	Oct-Dec
4111	LAKSHADWEEP	2011	5.1	2.8	3.1	85.9	107.2	153.6	350.2	254.0	255.2	117.4	184.3	14.9	1533.7	7.9	196.2	1013.0	316.6
4112	LAKSHADWEEP	2012	19.2	0.1	1.6	76.8	21.2	327.0	231.5	381.2	179.8	145.9	12.4	8.8	1405.5	19.3	99.6	1119.5	167.1
4113	LAKSHADWEEP	2013	26.2	34.4	37.5	5.3	88.3	426.2	296.4	154.4	180.0	72.8	78.1	26.7	1426.3	60.6	131.1	1057.0	177.6
4114	LAKSHADWEEP	2014	53.2	16.1	4.4	14.9	57.4	244.1	116.1	466.1	132.2	169.2	59.0	62.3	1395.0	69.3	76.7	958.5	290.5
4115	LAKSHADWEEP	2015	2.2	0.5	3.7	87.1	133.1	296.6	257.5	146.4	160.4	165.4	231.0	159.0	1642.9	2.7	223.9	860.9	555.4

In [83]:

```
df.shape
```

Out[83]:

```
(4116, 19)
```

In [84]:

```
df.describe
```

Out[84]:

```
<bound method NDFrame.describe of                     SUBDIVISION  YEAR   JAN    FEB   MAR    APR    MAY    JUN  \
0     ANDAMAN & NICOBAR ISLANDS  1901  49.2   87.1  29.2    2.3  528.8  517.5   
1     ANDAMAN & NICOBAR ISLANDS  1902   0.0  159.8  12.2    0.0  446.1  537.1   
2     ANDAMAN & NICOBAR ISLANDS  1903  12.7  144.0   0.0    1.0  235.1  479.9   
3     ANDAMAN & NICOBAR ISLANDS  1904   9.4   14.7   0.0  202.4  304.5  495.1   
4     ANDAMAN & NICOBAR ISLANDS  1905   1.3    0.0   3.3   26.9  279.5  628.7   
...                         ...   ...   ...    ...   ...    ...    ...    ...   
4111                LAKSHADWEEP  2011   5.1    2.8   3.1   85.9  107.2  153.6   
4112                LAKSHADWEEP  2012  19.2    0.1   1.6   76.8   21.2  327.0   
4113                LAKSHADWEEP  2013  26.2   34.4  37.5    5.3   88.3  426.2   
4114                LAKSHADWEEP  2014  53.2   16.1   4.4   14.9   57.4  244.1   
4115                LAKSHADWEEP  2015   2.2    0.5   3.7   87.1  133.1  296.6   

        JUL    AUG    SEP    OCT    NOV    DEC  ANNUAL  Jan-Feb  Mar-May  \
0     365.1  481.1  332.6  388.5  558.2   33.6  3373.2    136.3    560.3   
1     228.9  753.7  666.2  197.2  359.0  160.5  3520.7    159.8    458.3   
2     728.4  326.7  339.0  181.2  284.4  225.0  2957.4    156.7    236.1   
3     502.0  160.1  820.4  222.2  308.7   40.1  3079.6     24.1    506.9   
4     368.7  330.5  297.0  260.7   25.4  344.7  2566.7      1.3    309.7   
...     ...    ...    ...    ...    ...    ...     ...      ...      ...   
4111  350.2  254.0  255.2  117.4  184.3   14.9  1533.7      7.9    196.2   
4112  231.5  381.2  179.8  145.9   12.4    8.8  1405.5     19.3     99.6   
4113  296.4  154.4  180.0   72.8   78.1   26.7  1426.3     60.6    131.1   
4114  116.1  466.1  132.2  169.2   59.0   62.3  1395.0     69.3     76.7   
4115  257.5  146.4  160.4  165.4  231.0  159.0  1642.9      2.7    223.9   

      Jun-Sep  Oct-Dec  
0      1696.3    980.3  
1      2185.9    716.7  
2      1874.0    690.6  
3      1977.6    571.0  
4      1624.9    630.8  
...       ...      ...  
4111   1013.0    316.6  
4112   1119.5    167.1  
4113   1057.0    177.6  
4114    958.5    290.5  
4115    860.9    555.4  

[4116 rows x 19 columns]>
```

In [85]:

```
df.info
```

Out[85]:

```
<bound method DataFrame.info of                      SUBDIVISION  YEAR   JAN    FEB   MAR    APR    MAY    JUN  
0     ANDAMAN & NICOBAR ISLANDS  1901  49.2   87.1  29.2    2.3  528.8  517.5   \
1     ANDAMAN & NICOBAR ISLANDS  1902   0.0  159.8  12.2    0.0  446.1  537.1
2     ANDAMAN & NICOBAR ISLANDS  1903  12.7  144.0   0.0    1.0  235.1  479.9
3     ANDAMAN & NICOBAR ISLANDS  1904   9.4   14.7   0.0  202.4  304.5  495.1
4     ANDAMAN & NICOBAR ISLANDS  1905   1.3    0.0   3.3   26.9  279.5  628.7
...                         ...   ...   ...    ...   ...    ...    ...    ...
4111                LAKSHADWEEP  2011   5.1    2.8   3.1   85.9  107.2  153.6
4112                LAKSHADWEEP  2012  19.2    0.1   1.6   76.8   21.2  327.0
4113                LAKSHADWEEP  2013  26.2   34.4  37.5    5.3   88.3  426.2
4114                LAKSHADWEEP  2014  53.2   16.1   4.4   14.9   57.4  244.1
4115                LAKSHADWEEP  2015   2.2    0.5   3.7   87.1  133.1  296.6

        JUL    AUG    SEP    OCT    NOV    DEC  ANNUAL  Jan-Feb  Mar-May  
0     365.1  481.1  332.6  388.5  558.2   33.6  3373.2    136.3    560.3   \
1     228.9  753.7  666.2  197.2  359.0  160.5  3520.7    159.8    458.3
2     728.4  326.7  339.0  181.2  284.4  225.0  2957.4    156.7    236.1
3     502.0  160.1  820.4  222.2  308.7   40.1  3079.6     24.1    506.9
4     368.7  330.5  297.0  260.7   25.4  344.7  2566.7      1.3    309.7
...     ...    ...    ...    ...    ...    ...     ...      ...      ...
4111  350.2  254.0  255.2  117.4  184.3   14.9  1533.7      7.9    196.2
4112  231.5  381.2  179.8  145.9   12.4    8.8  1405.5     19.3     99.6
4113  296.4  154.4  180.0   72.8   78.1   26.7  1426.3     60.6    131.1
4114  116.1  466.1  132.2  169.2   59.0   62.3  1395.0     69.3     76.7
4115  257.5  146.4  160.4  165.4  231.0  159.0  1642.9      2.7    223.9

      Jun-Sep  Oct-Dec  
0      1696.3    980.3  
1      2185.9    716.7  
2      1874.0    690.6  
3      1977.6    571.0  
4      1624.9    630.8  
...       ...      ...  
4111   1013.0    316.6  
4112   1119.5    167.1  
4113   1057.0    177.6  
4114    958.5    290.5  
4115    860.9    555.4  

[4116 rows x 19 columns]>
```

In [86]:

```
df.isnull().sum()
```

Out[86]:

```
SUBDIVISION     0
YEAR            0
JAN             4
FEB             3
MAR             6
APR             4
MAY             3
JUN             5
JUL             7
AUG             4
SEP             6
OCT             7
NOV            11
DEC            10
ANNUAL         26
Jan-Feb         6
Mar-May         9
Jun-Sep        10
Oct-Dec        13
dtype: int64
```

In [87]:

```
df.fillna(method="ffill",inplace=True)
```

In [88]:

```
df.isnull().sum()
```

Out[88]:

```
SUBDIVISION    0
YEAR           0
JAN            0
FEB            0
MAR            0
APR            0
MAY            0
JUN            0
JUL            0
AUG            0
SEP            0
OCT            0
NOV            0
DEC            0
ANNUAL         0
Jan-Feb        0
Mar-May        0
Jun-Sep        0
Oct-Dec        0
dtype: int64
```

In [89]:

```
df['YEAR'].value_counts()
```

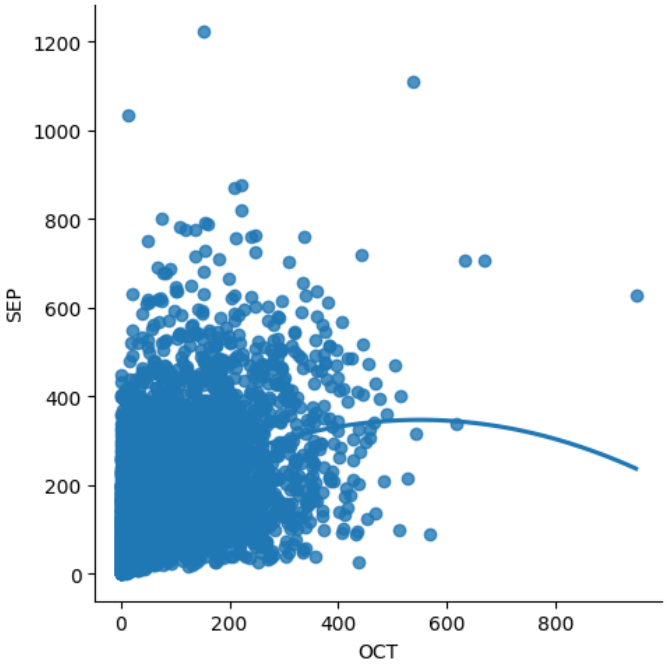
Out[89]:

```
YEAR
1963    36
2002    36
1976    36
1975    36
1974    36
        ..
1915    35
1918    35
1954    35
1955    35
1909    34
Name: count, Length: 115, dtype: int64
```

3)Exploratory Data Analysis

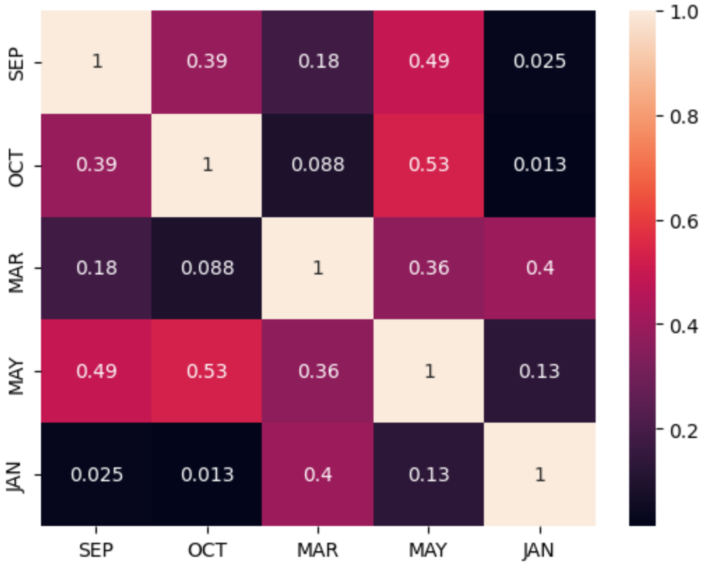
In [102]:

```
sns.lmplot(x='OCT',y='SEP',order=2,data=df,ci=None)
plt.show()
```



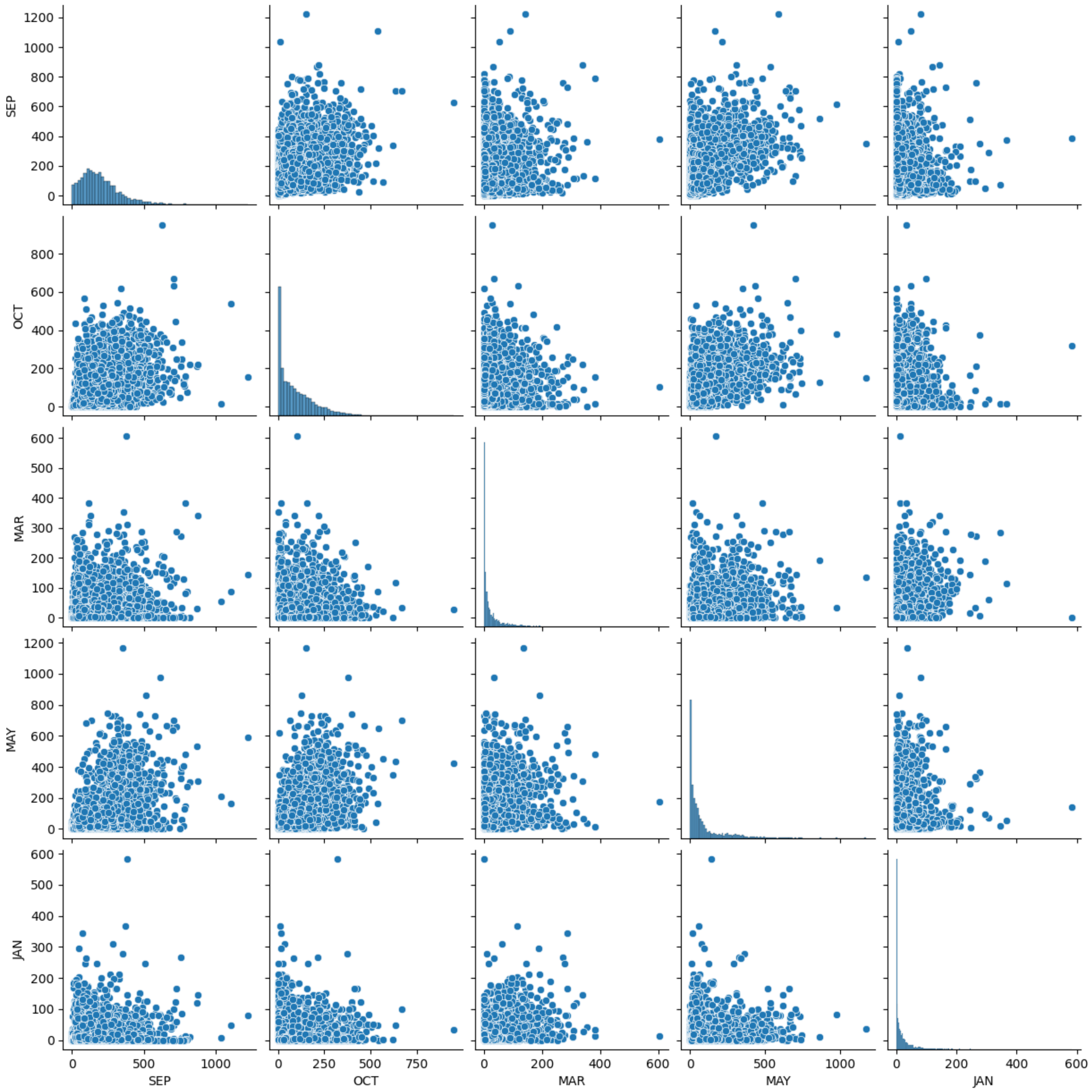
In [103]:

```
df=df[['SEP','OCT','MAR','MAY','JAN']]
sns.heatmap(df.corr(),annot=True)
plt.show()
```



In [104]:

```
sns.pairplot(df)
plt.show()
```



4)Training the Model

In [105]:

```
x=np.array(df['SEP']).reshape(-1,1)
y=x=np.array(df['OCT']).reshape(-1,1)
```

In [106]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30)
```

In [107]:

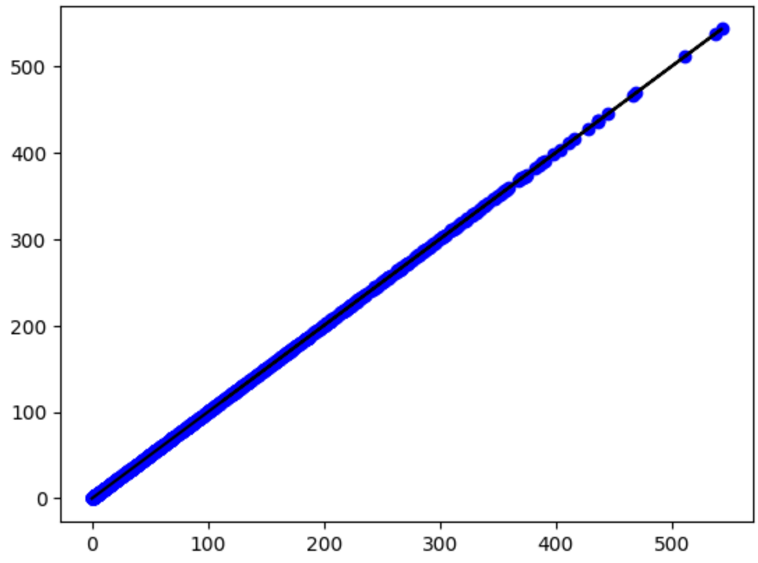
```
lin=LinearRegression()
lin.fit(x_train,y_train)
print(lin.score(x_train,y_train))
```

```
1.0
```

5)Exploring Results

In [108]:

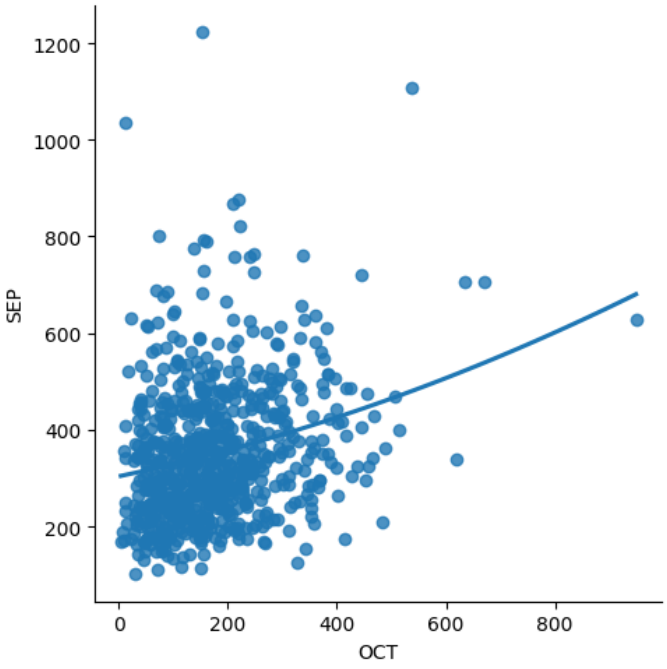
```
y_pred=lin.predict(x_test)
plt.scatter(x_test,y_test,color='blue')
plt.plot(x_test,y_pred,color='black')
plt.show()
```



7)Working with subset of data

In [109]:

```
df700=df[:][:700]
sns.lmplot(x='OCT',y='SEP',order=2,ci=None,data=df700)
plt.show()
```



In [110]:

```
df700.fillna(method='ffill',inplace=True)
```

In [111]:

```
x=np.array(df700['OCT']).reshape(-1,1)
y=x=np.array(df700['SEP']).reshape(-1,1)
```

In [112]:

```
df700.dropna(inplace=True)
```

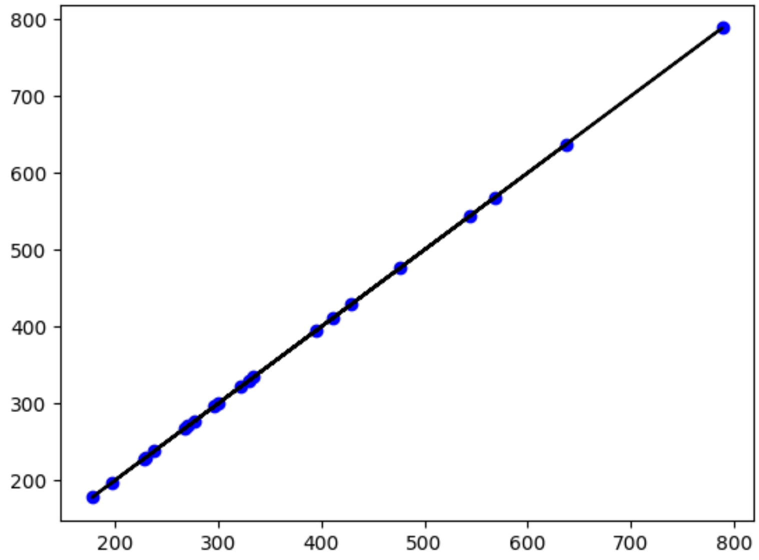
In [113]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.03)
lr=LinearRegression()
lr.fit(x_train,y_train)
print(lr.score(x_test,y_test))
```

```
1.0
```

In [114]:

```
y_pred=lr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [115]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [116]:

```
lr=LinearRegression()
lr.fit(x_train,y_train)
y_pred=lr.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 1.0
```

Accuracy for the Linear Regression is: 1.0

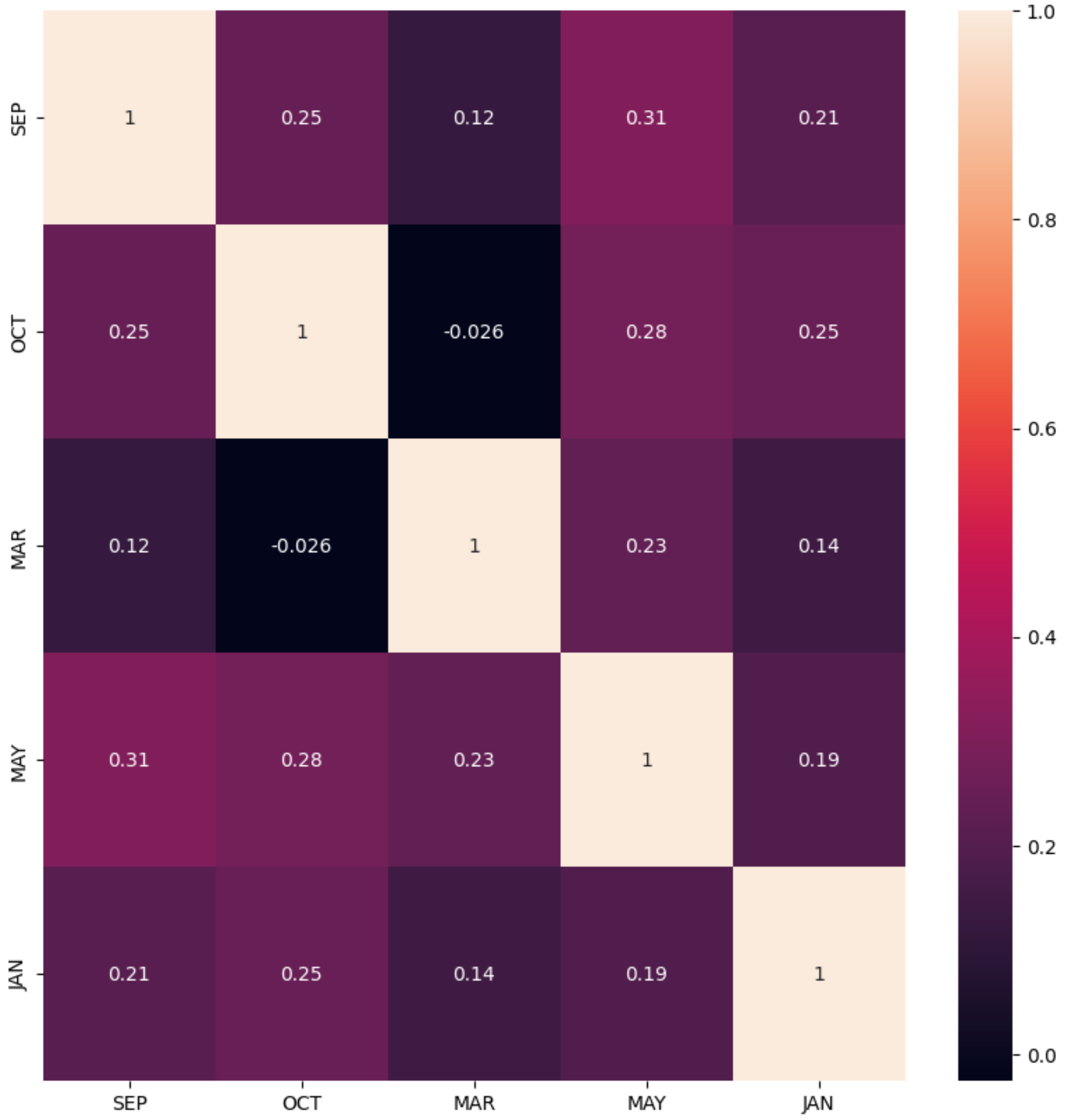
Ridge Regression

In [117]:

```
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import StandardScaler
```

In [118]:

```
plt.figure(figsize=(10,10))
sns.heatmap(df700.corr(),annot=True)
plt.show()
```



In [120]:

```
features=df.columns[0:5]
target=df.columns[-5]
```

In [121]:

```
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=1)
print("The dimension of X_train is {}".format(x_train.shape))
print("The dimension of X_test is {}".format(x_test.shape))
```

```
The dimension of X_train is (2881, 5)
The dimension of X_test is (1235, 5)
```

In [122]:

```
lr = LinearRegression()
#Fit model
lr.fit(x_train, y_train)
#predict
actual = y_test
train_score_lr = lr.score(x_train, y_train)
test_score_lr = lr.score(x_test, y_test)
print("\nLinear Regression Model:\n")
print("The train score for lr model is {}".format(train_score_lr))
print("The test score for lr model is {}".format(test_score_lr))
```

```

Linear Regression Model:

The train score for lr model is 1.0
The test score for lr model is 1.0
```

In [123]:

```
ridgeReg = Ridge(alpha=10)
ridgeReg.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ridge = ridgeReg.score(x_train, y_train)
test_score_ridge = ridgeReg.score(x_test, y_test)
print("\nRidge Model:\n")
print("The train score for ridge model is {}".format(train_score_ridge))
print("The test score for ridge model is {}".format(test_score_ridge))
```

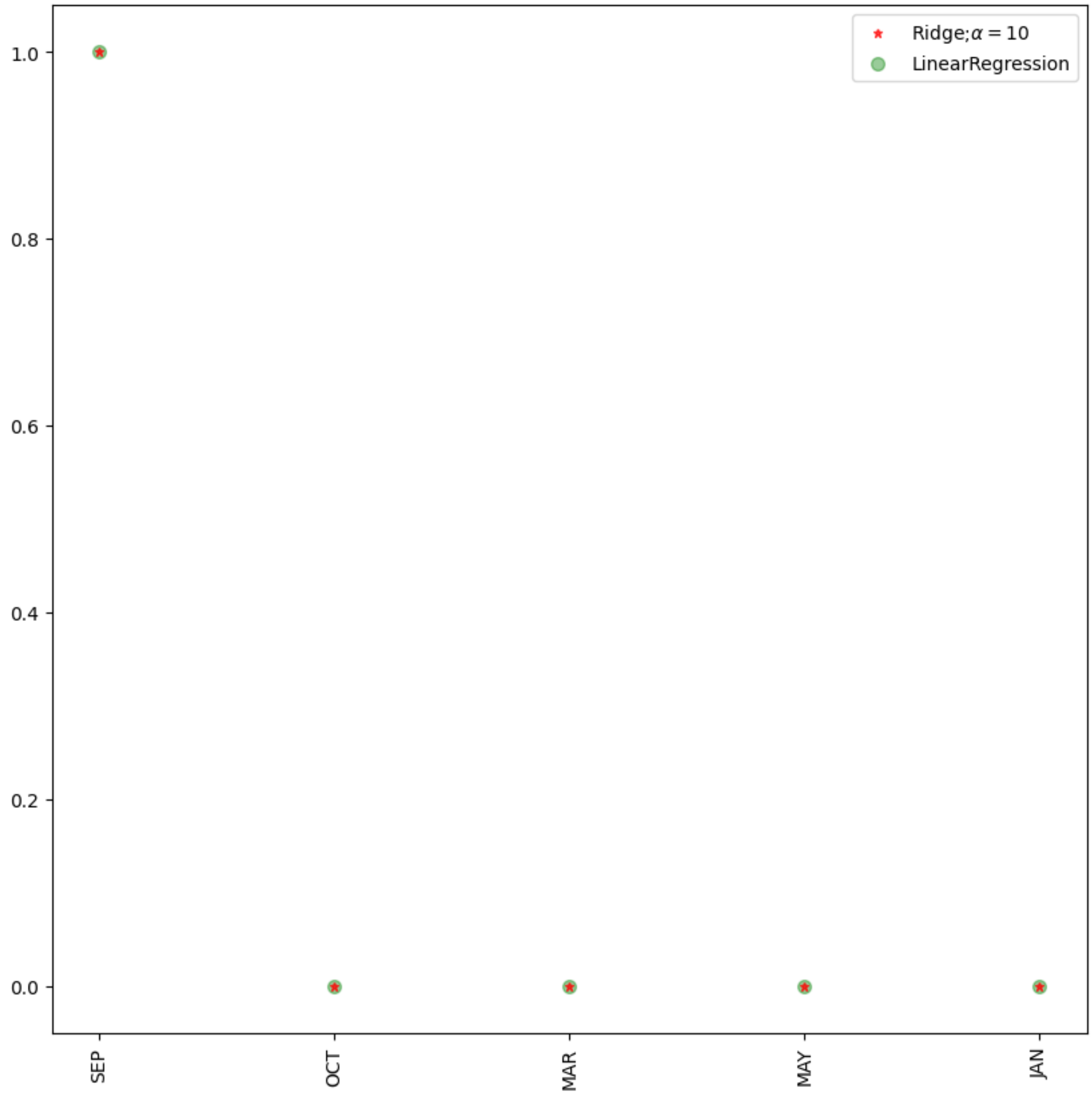
```

Ridge Model:

The train score for ridge model is 0.9999999999999525
The test score for ridge model is 0.9999999999999503
```

In [124]:

```
plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker="*",markersize=5,color='red',label=r'Ridge;$\alpha=10$',zorder=7
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker="o",markersize=7,color='green',label='LinearRegression')
plt.xticks(rotation=90)
plt.legend()
plt.show()
```



Accuracy for the Ridge Regression is: 0.99

Lasso Regression

In [125]:

```
#Importing libraries
lasso= Lasso(alpha=10)
lasso.fit(x_train,y_train)
#train and test scorefor ridge regression
train_score_ls = lasso.score(x_train, y_train)
test_score_ls= lasso.score(x_test, y_test)
print("\nLasso Model:\n")
print("The train score for lasso model is {}".format(train_score_ls))
print("The test score for lasso model is {}".format(test_score_ls))
```

```
Lasso Model:

The train score for lasso model is 0.9999997141681314
The test score for lasso model is 0.9999997138753469
```

In [126]:

```
plt.figure(figsize=(10,10))
```

Out[126]:

```
<Figure size 1000x1000 with 0 Axes>
```

```
<Figure size 1000x1000 with 0 Axes>
```

In [127]:

```
from sklearn.linear_model import LassoCV
```

In [128]:

```
from sklearn.linear_model import RidgeCV
#cross validation
ridge_cv=RidgeCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(ridge_cv.score(x_train,y_train))
print(ridge_cv.score(x_test,y_test))
```

```
0.999999042457621
0.9999990959631722
```

In [131]:

```
#using the linear cv model
from sklearn.linear_model import LassoCV
#cross validation
lasso_cv=LassoCV(alphas =[0.0001,0.001,0.01,0.1,1,10]).fit(x_train,y_train)
#score
print(lasso_cv.score(x_train,y_train))
print(lasso_cv.score(x_test,y_test))
```

```
1.0
1.0
```

Accuracy for the Lasso Regression is: 1.0

Elastic Net

In [134]:

```
from sklearn.linear_model import ElasticNet
```

In [135]:

```
el=ElasticNet()
el.fit(x_train,y_train)
print(el.coef_)
print(el.intercept_)
el.score(x,y)
```

```
[0.99994654 0.         0.         0.         0.        ]
0.010628023653254104
```

Out[135]:

```
0.9999999971415823
```

In [136]:

```
y_pred_elastic=el.predict(x_train)
```

In [137]:

```
mean_squared_error=np.mean((y_pred_elastic-y_train)**2)
print(mean_squared_error)
```

```
5.346038791995705e-05
```

Accuracy for the Elastic Net is: 0.99

Conclusion

The given dataset is 'Rain Fall Prediction'.Here we need to find the best fit model.As per the given data set I had applyed different types ofmodels...in which different type of models got different type of accuracies

Accuracy for the Linear Regression is: 1.0

The accuracy of the Ridge Model is 0.99

The accuracy of the Lasso Model is 1.0

The accuracy of the ElasticNet Regression is 0.99

comparing to all the models,Ridge Regression got the Highest Accuracy

In [ ]:

```

```